Informed Search

CS4804 Introduction to Artificial Intelligence
Virginia Tech

function TREE-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

loop do:

if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the solution

expand the chosen node, adding the resulting nodes to frontier

function TREE-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

loop do:

if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the solution

expand the chosen node, adding the resulting nodes to frontier

FIFO: Breadth-first
LIFO: Depth-first

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do:

if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier
only if not in the frontier or explored set

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do:

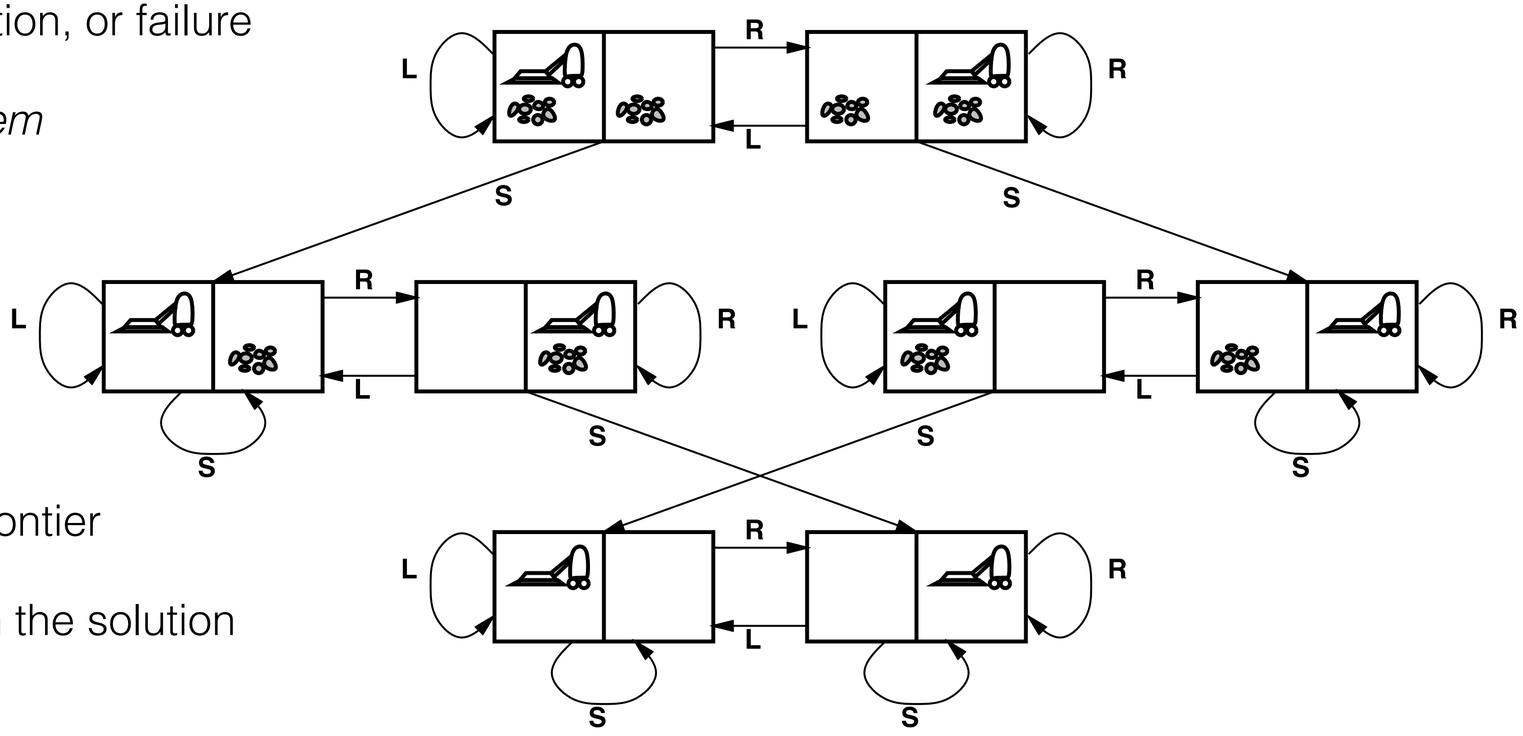
if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier
only if not in the frontier or explored set



function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do:

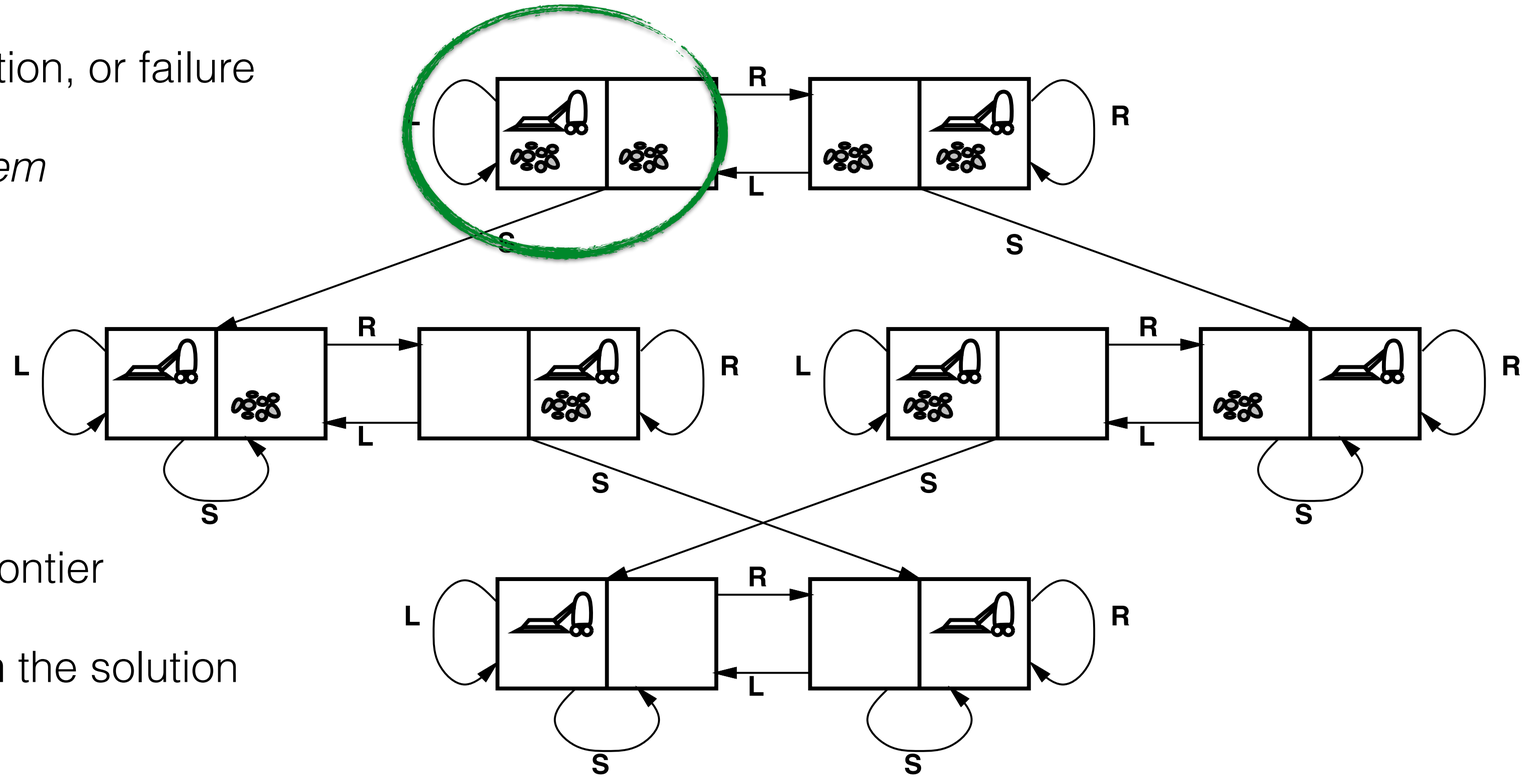
if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier
only if not in the frontier or explored set



function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do:

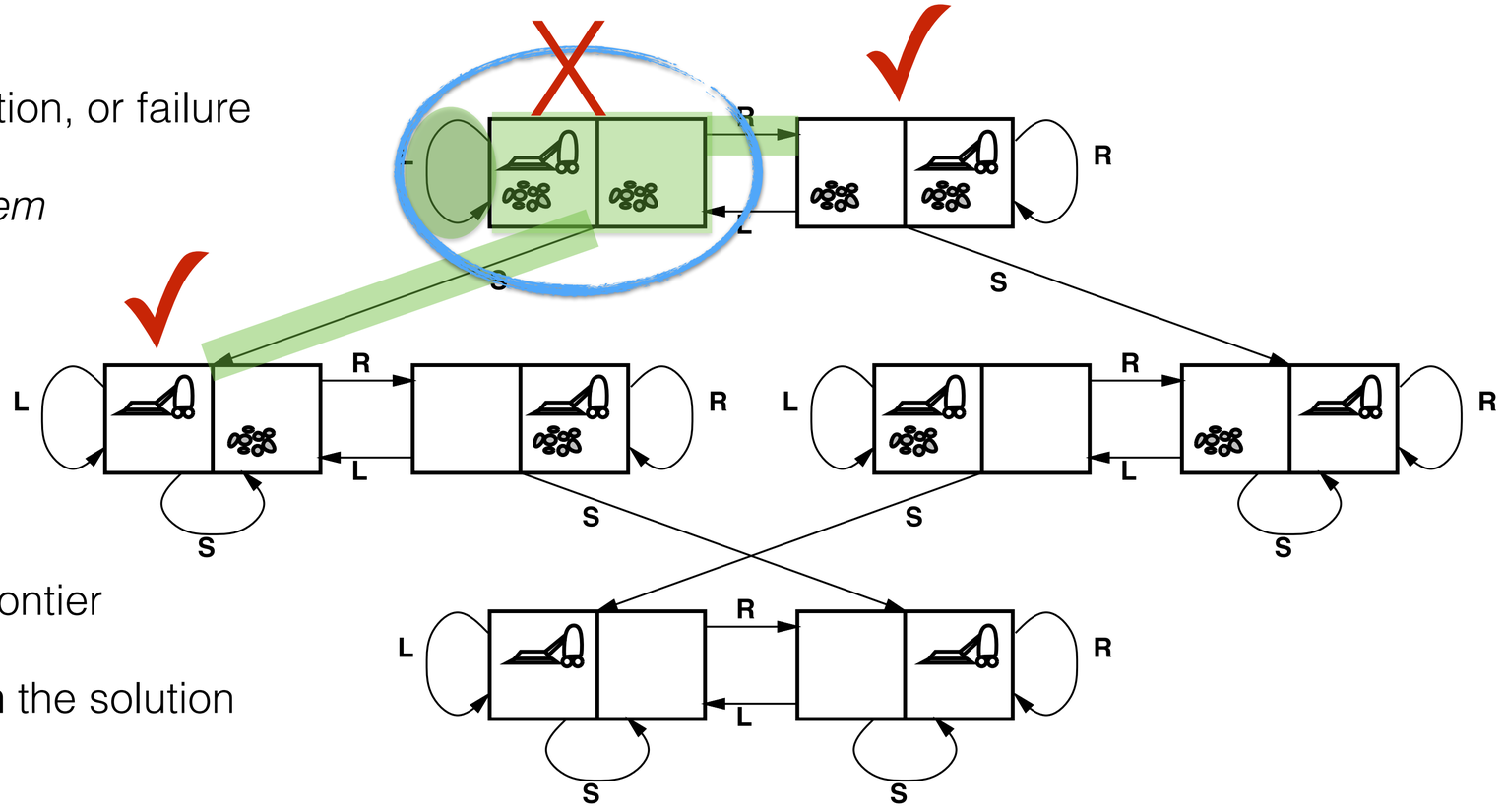
if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier
only if not in the frontier or explored set



function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do:

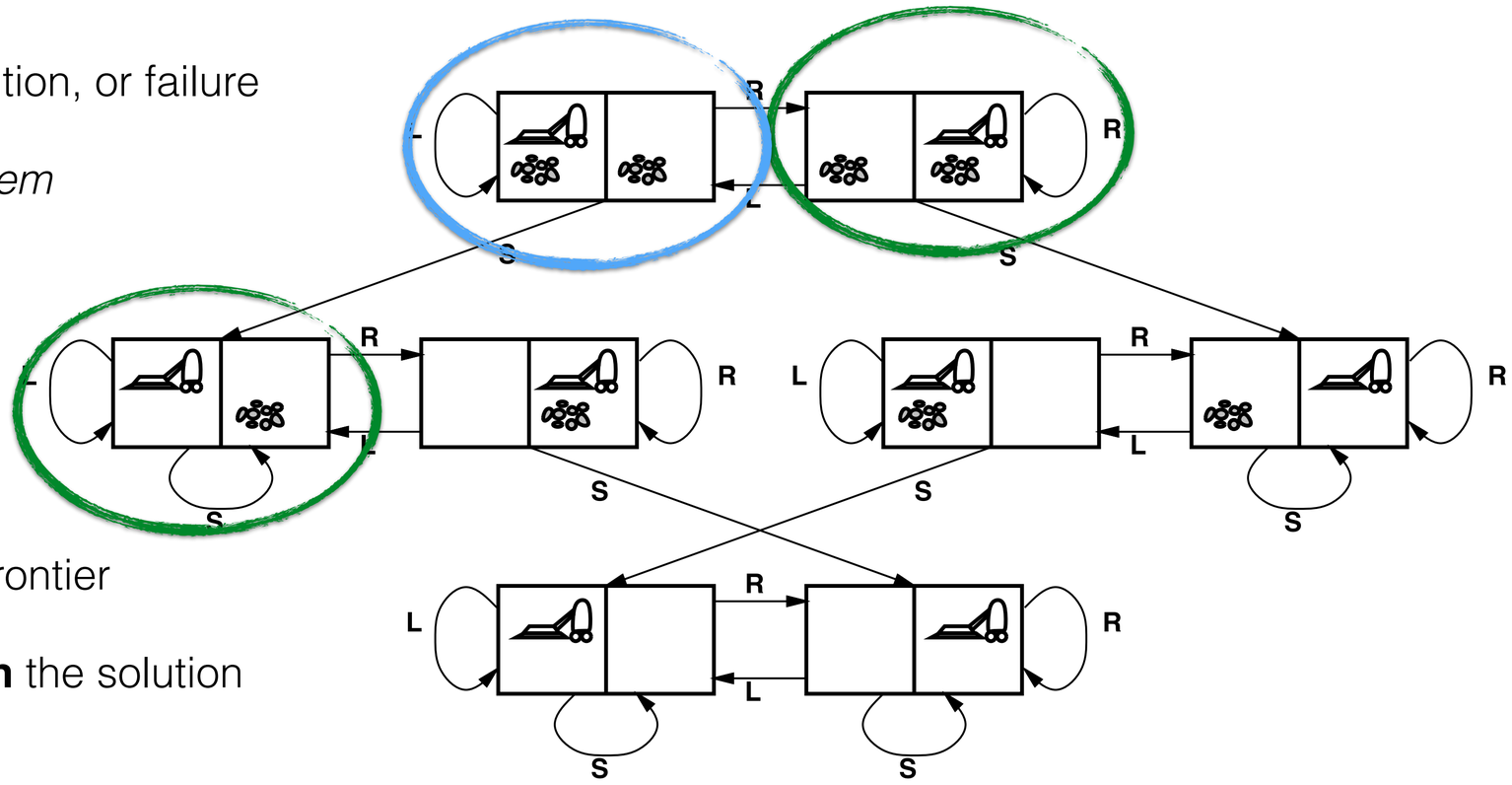
if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier
only if not in the frontier or explored set



function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do:

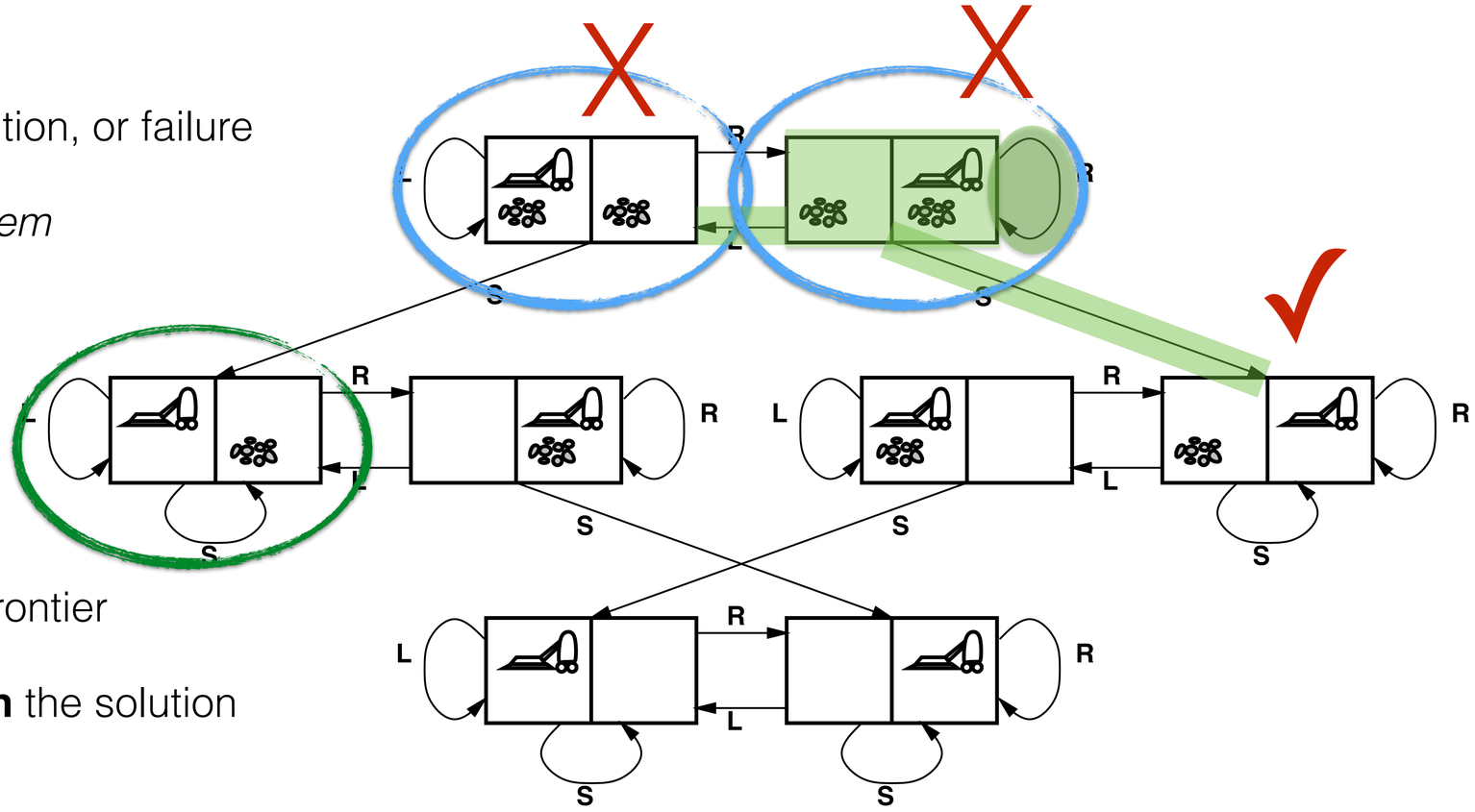
if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to frontier
only if not in the frontier or explored set



function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do:

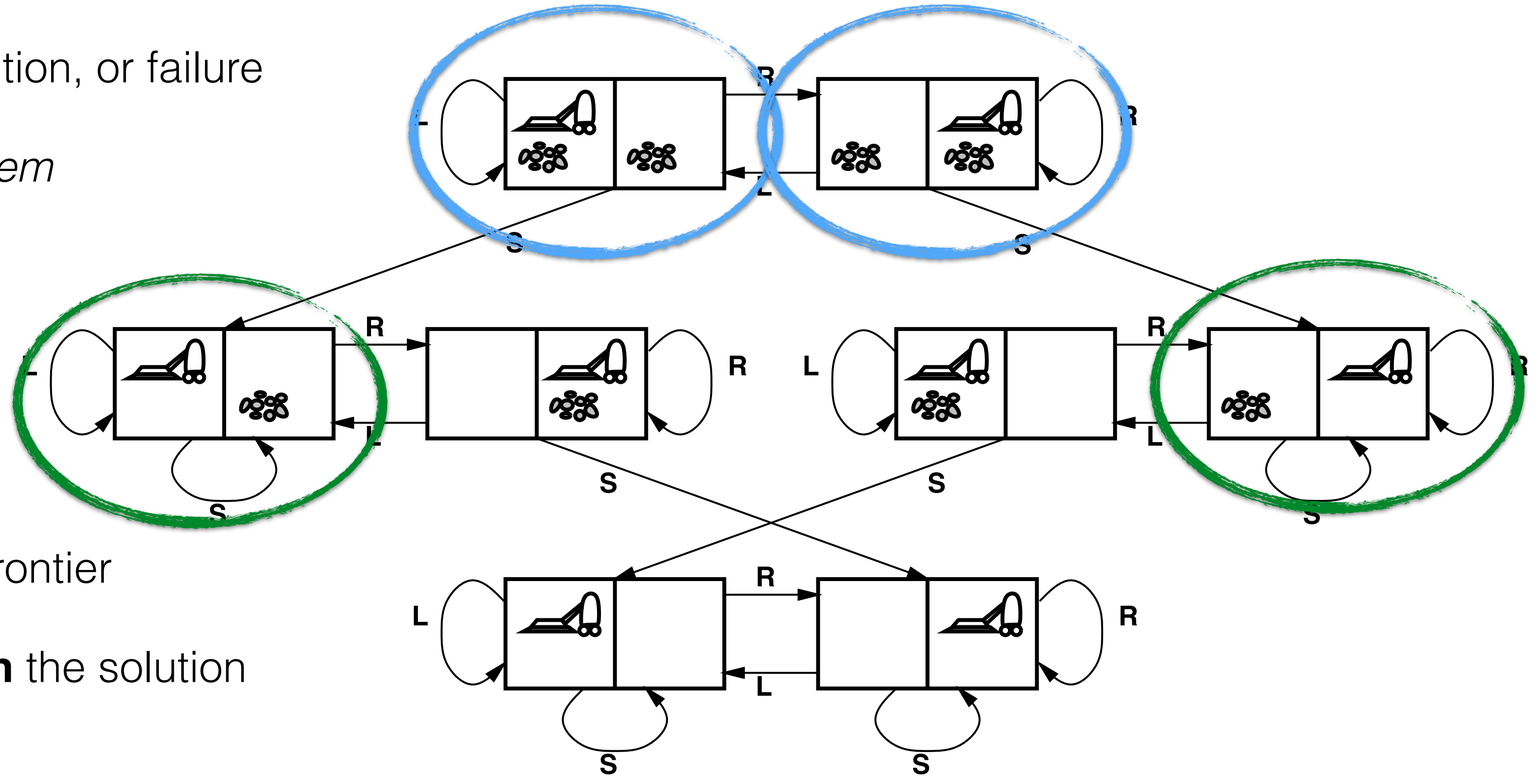
if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the solution

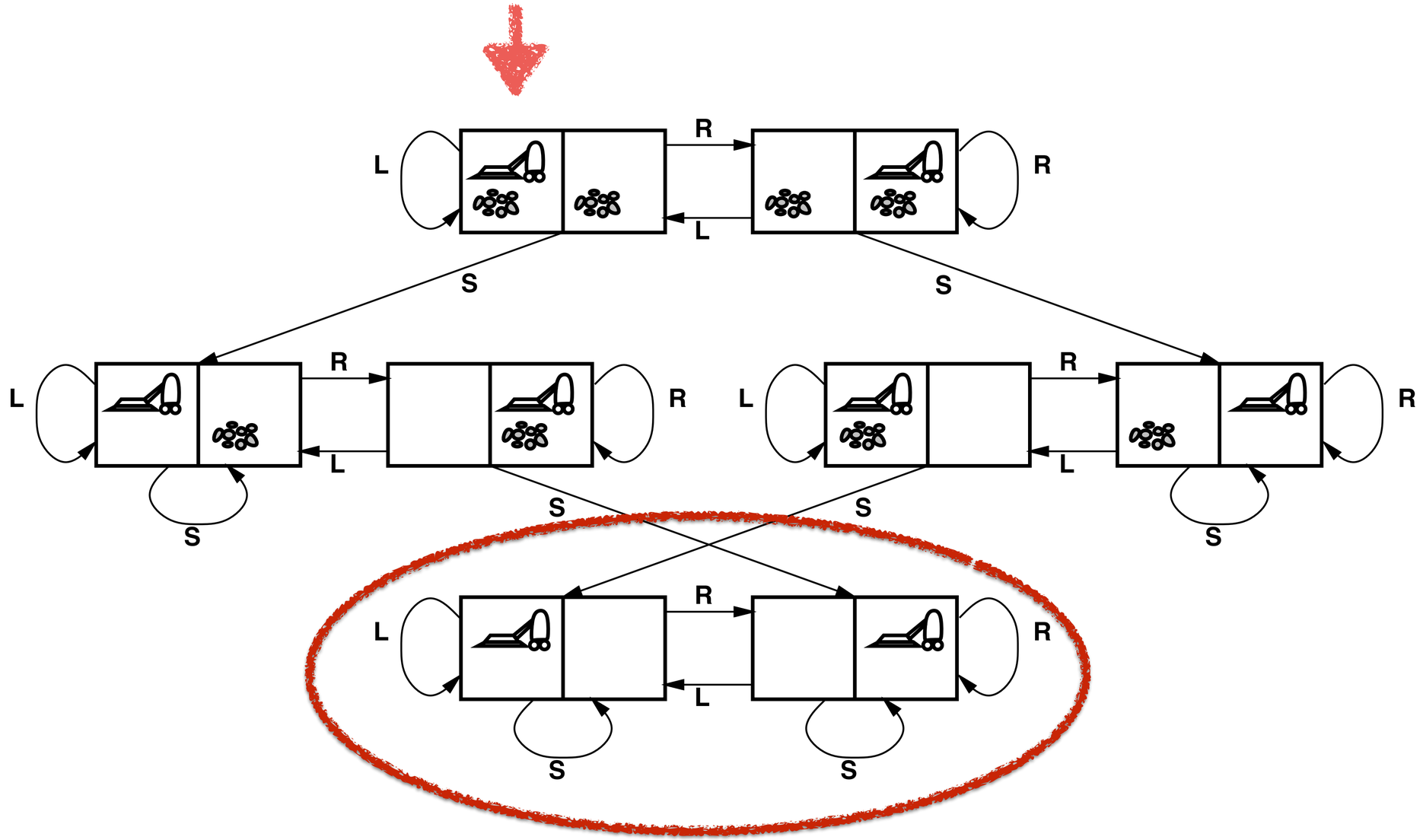
add the node to the explored set

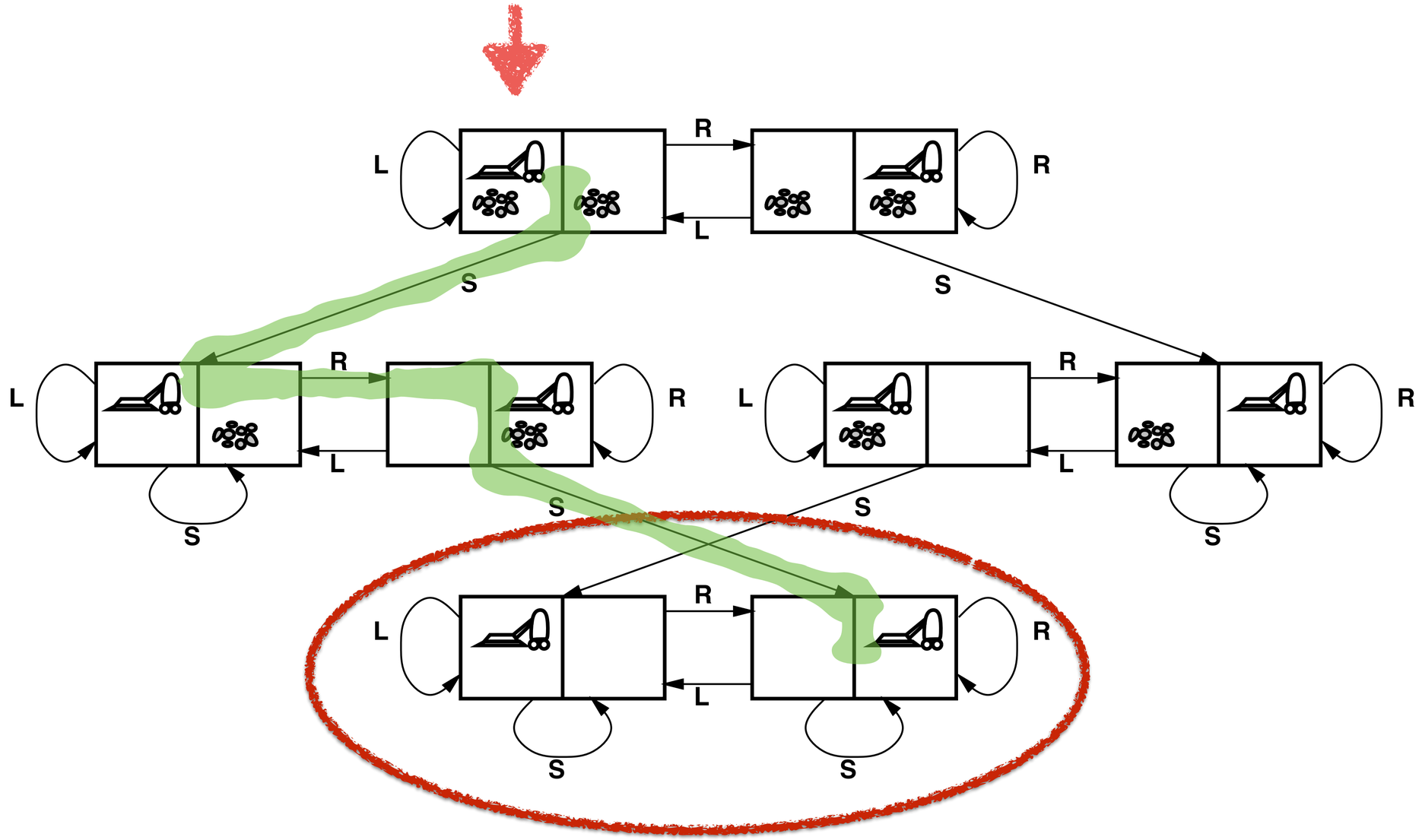
expand the chosen node, adding the resulting nodes to frontier
only if not in the frontier or explored set

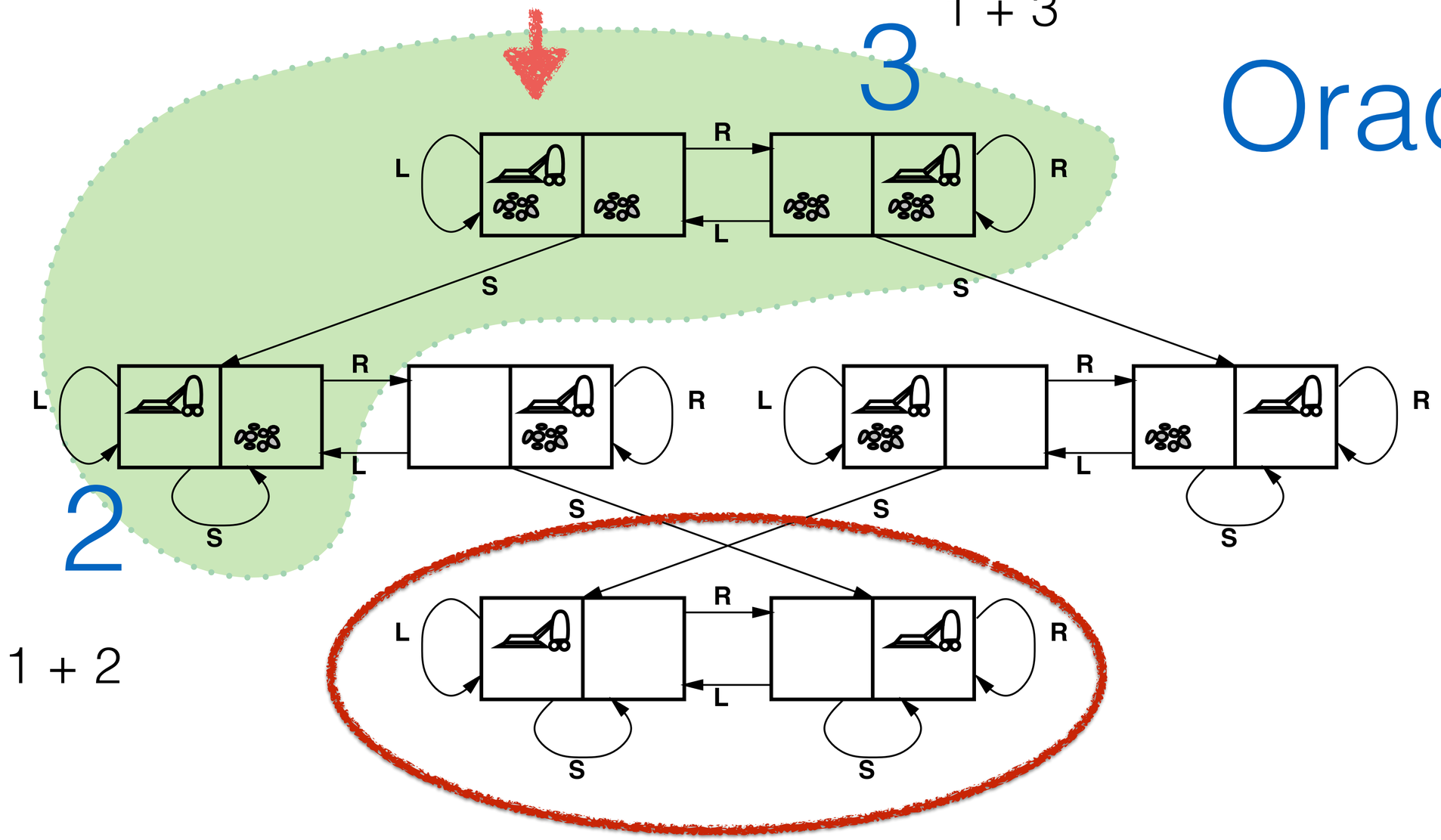


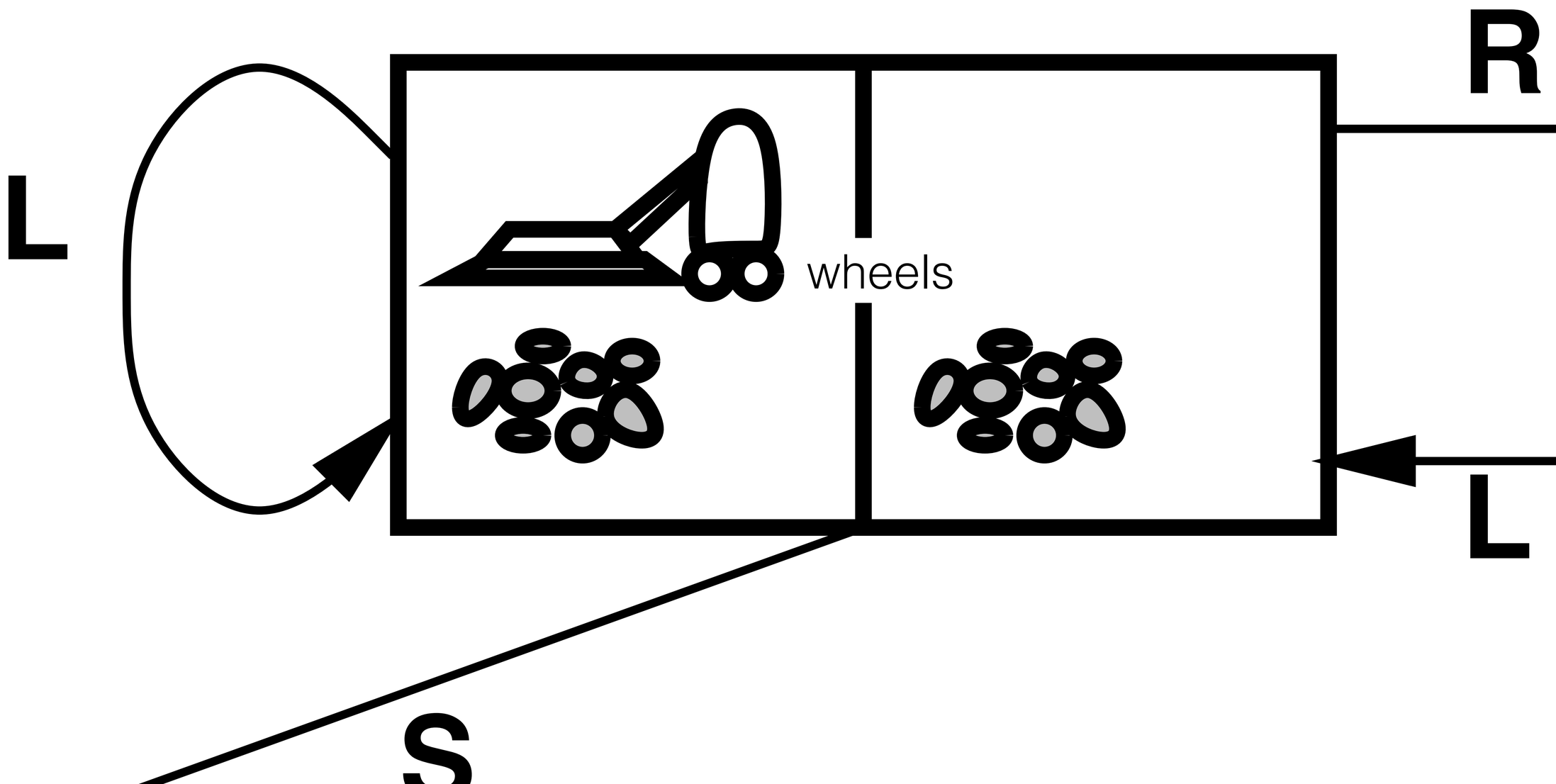
Search Algorithms

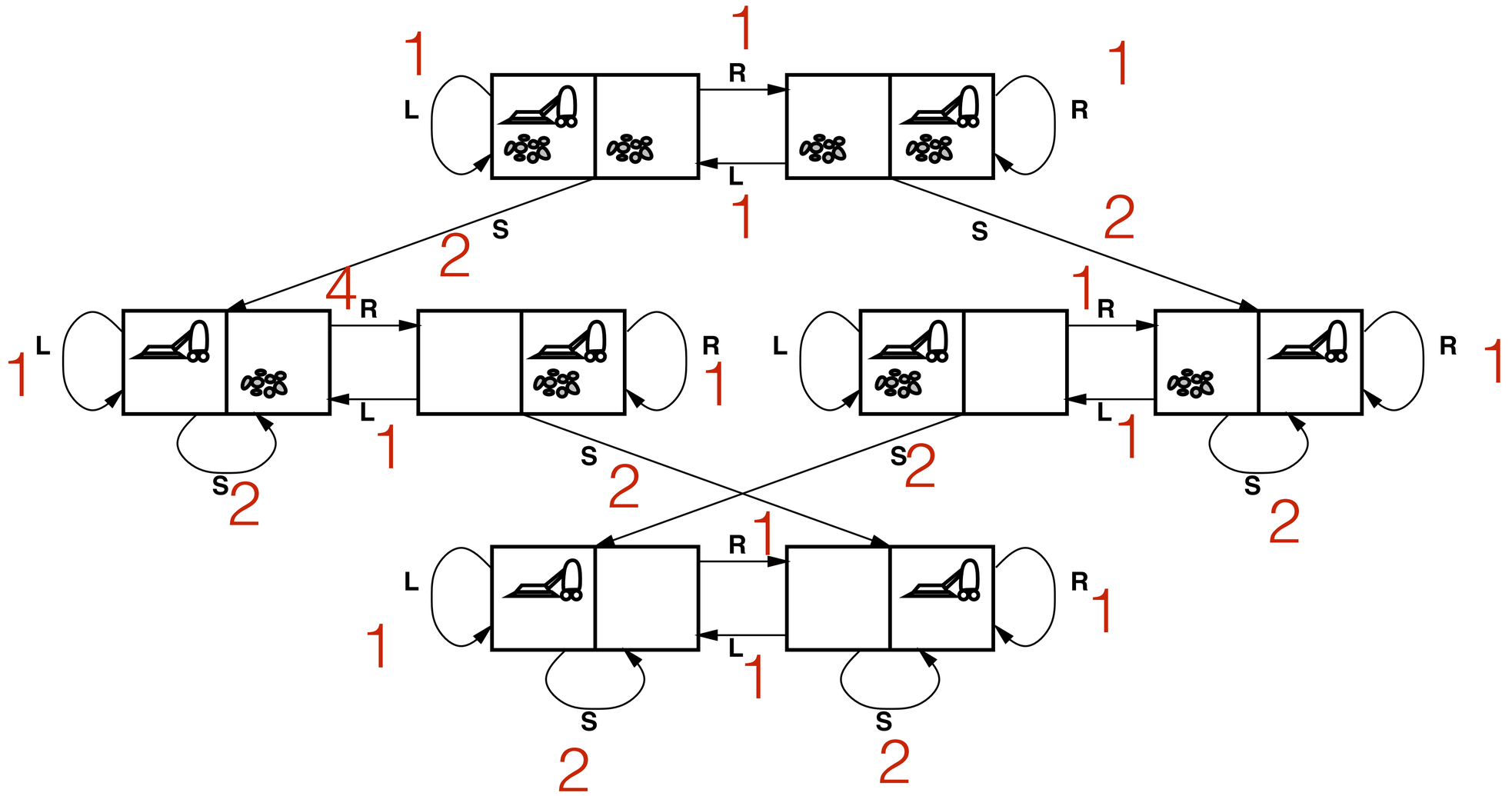
- Breadth-first family
  - Breadth-first search
  - Uniform-cost search
- Depth-first family
  - Depth-first search
  - Depth-limited search
  - Iterative-deepening search

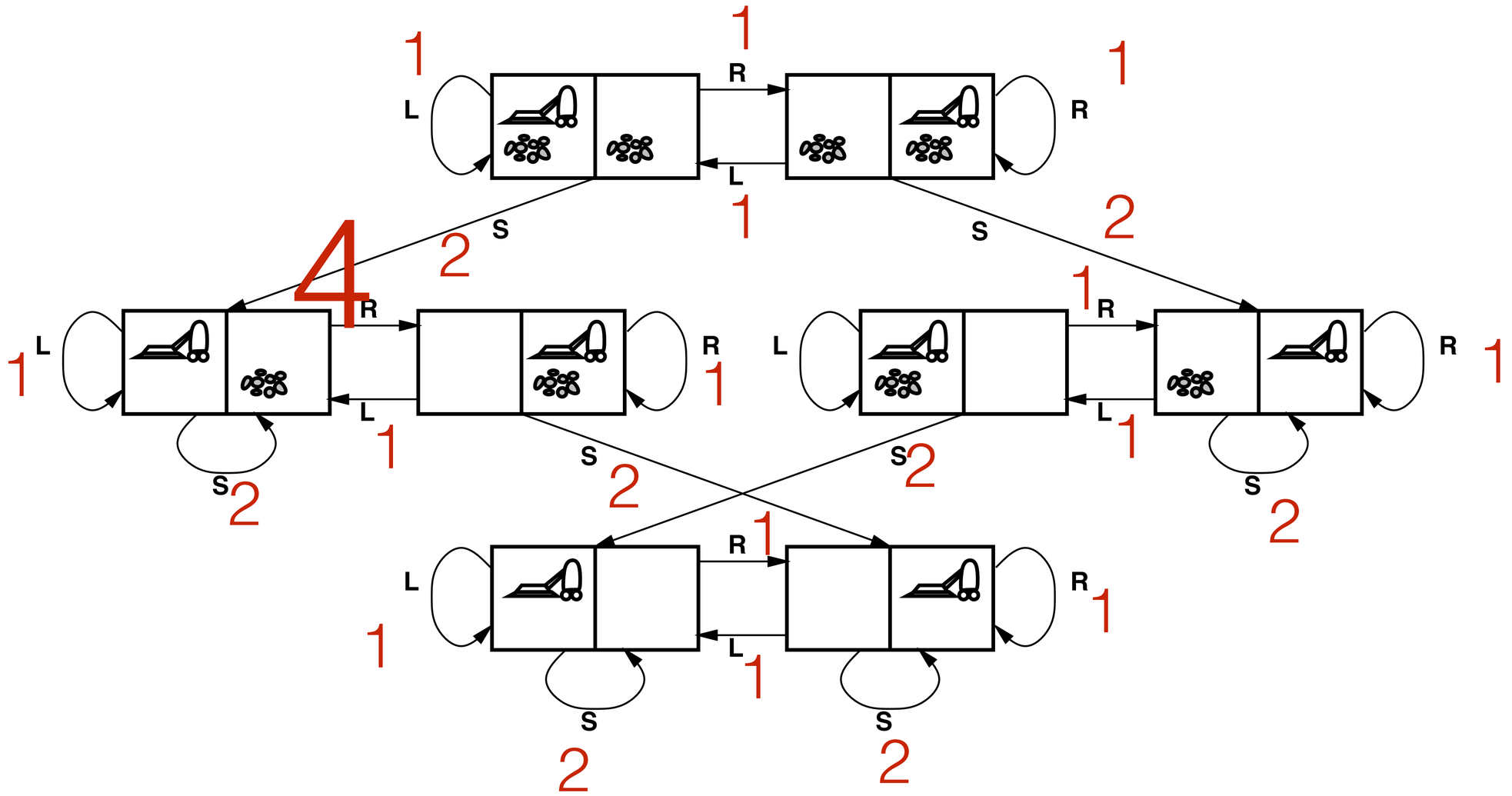

L
R
L
R
S
S
L
R
L
R
S
S
L
R
L
R
S
S
L
R
L
R
S
S

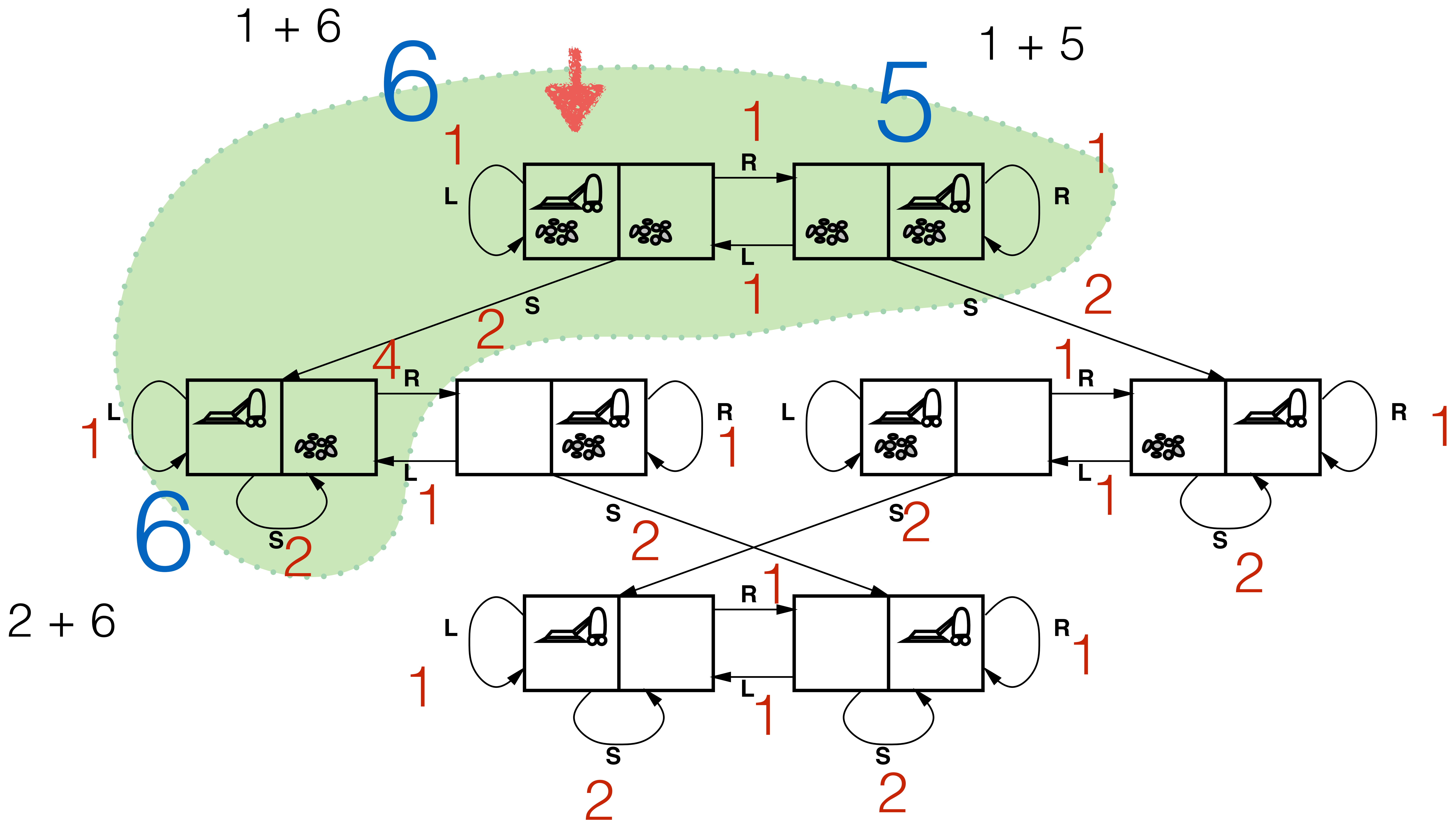

R
L
R
L
S
S
R
L
R
L
S
S
R
L
R
L
S
S
R
L
R
L
S
S


1 + 3
3
Oracle
R
L
L
R
S
S
L
R
R
L
L
R
S
S
S
2
S
1 + 2
R
L
L
R
S
S


R
L
wheels
L
S

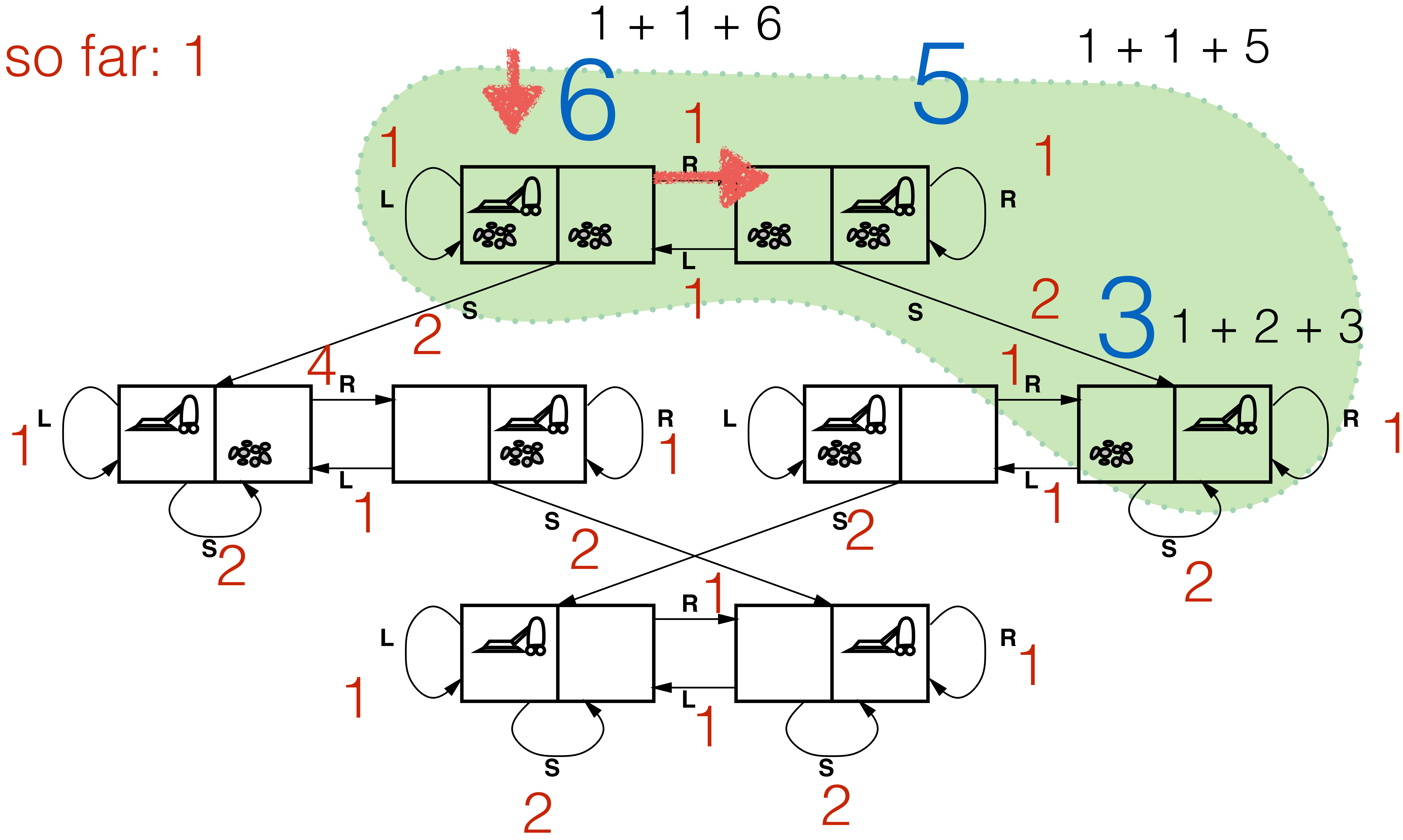

1
1
R
1
L
R
L
1
S
2
S
2
4
R
1
R
L
1
L
R
1
L
1
S
2
S
2
S
2
1
R
L
R
1
L
1
1
S
2
S
2


4
L
R
S
1
2

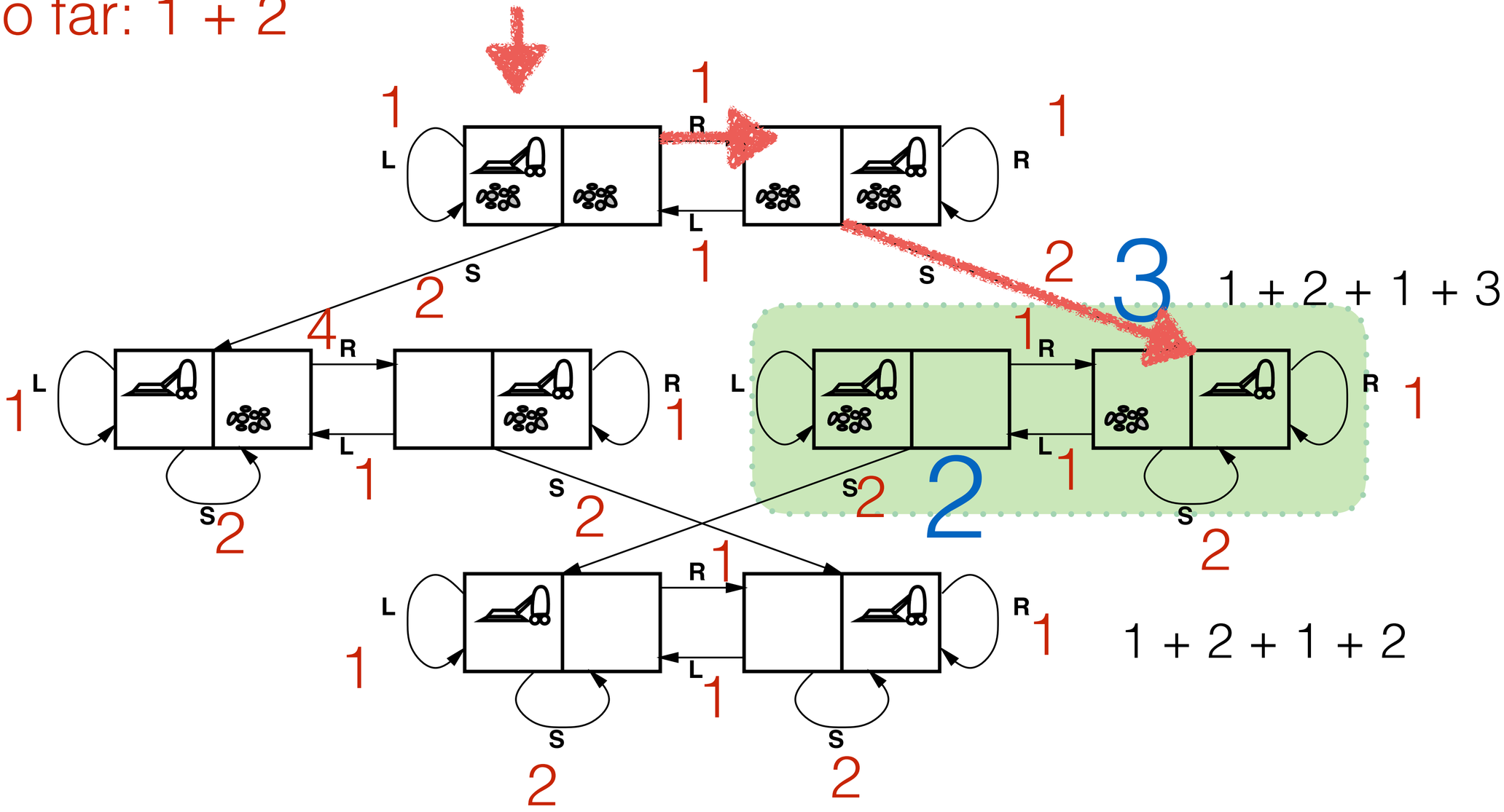


1 + 6
1 + 5
2 + 6
6
5
6
L
R
S

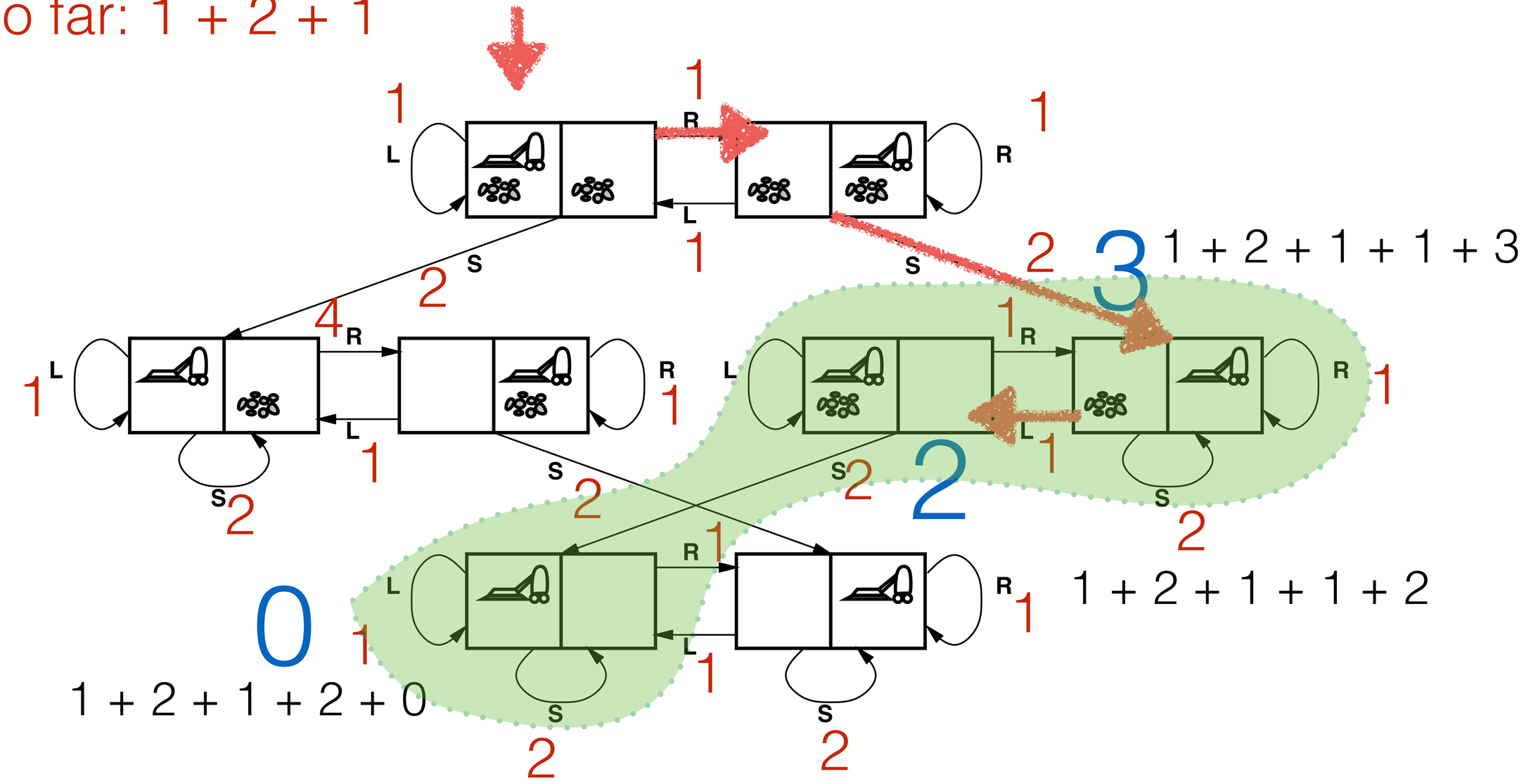
cost so far: 1



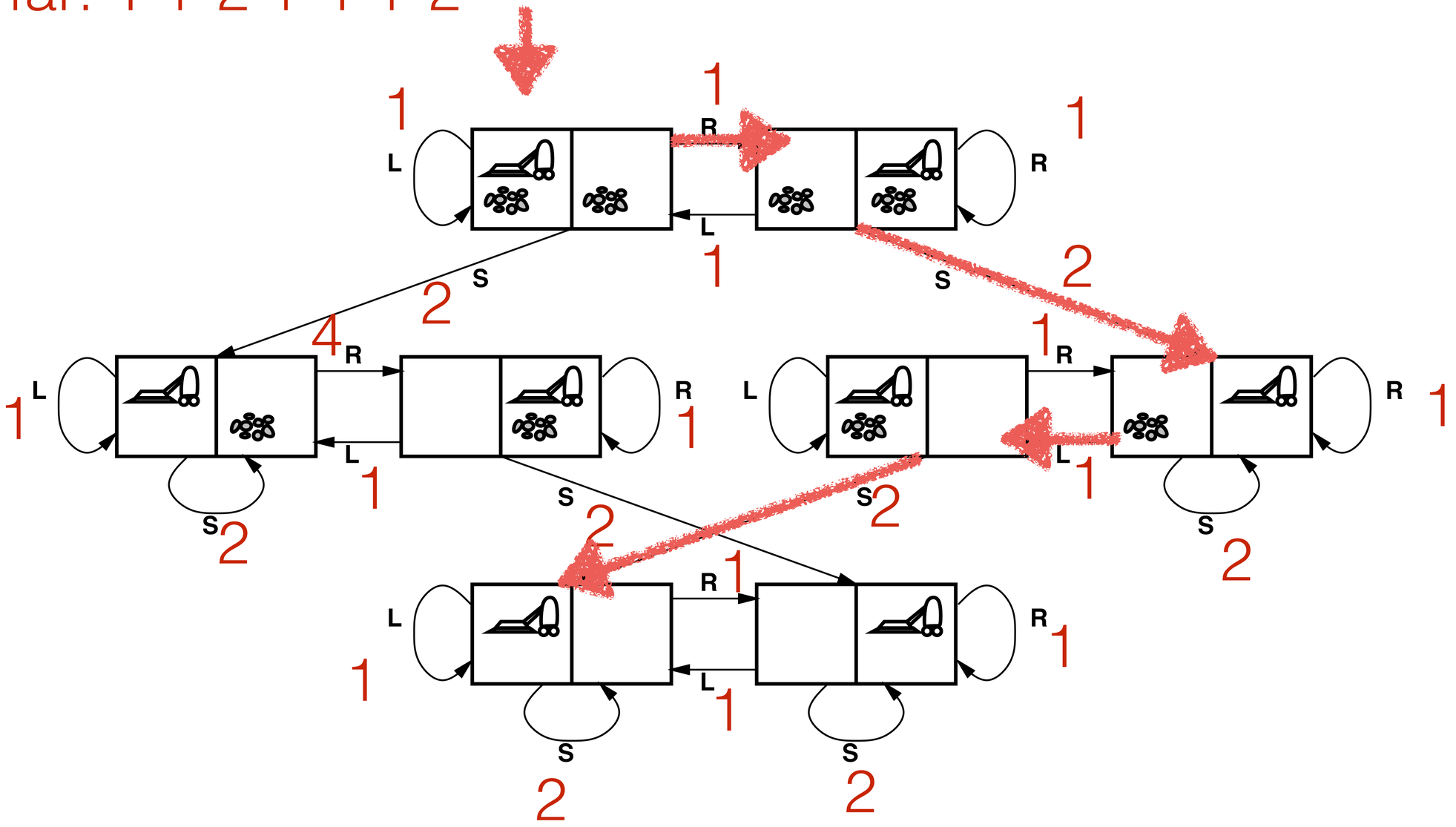
cost so far: 1 + 2



cost so far: 1 + 2 + 1



cost so far: 1 + 2 + 1 + 2



A* Search

- Expand node in frontier with best evaluation function score **f(n)**
  - **f(n) = g(n) + h(n)**
  - **g(n) :=** cost to get from initial state to **n**
  - **h(n) :=** heuristic estimate of cost to get from **n** to goal
- Completeness, optimality, and time & space depend on **h**